Mars Orbit Module

Prakhar Gupta, 17855, MTech AI

August 27, 2021

Abstract

Created a function to determine the parameters to best fit the mars orbital. Discretized search is done across all parameters for all the 5 questions. These parameters gives the best goodness of fit of the Mars orbital. For each 5 questions, the corresponding 5 functions were created. All these 5 functions call 1 common function "minimize(comb_list,oppositions)", whose task is to obtain the best parameters provided an opposition data and an iterable list that contains all the possible parameter combinations. Moreover the best parameters obtained from each question is taken as an initial guess for the following question. Code Architecture is explained in detail in the upcoming sections.

1 Procedure

Given polar coordinate of Equant (Eq) is (e1,e2+z) $\implies$ Cartesian cooridnate Eq (h,k),

$$h = e1cos(e2 + z), k = e1sin(e2 + z)$$

.
On a random spoke from Eq at an angle $\phi = z + st$, we obtain a points (let) P1 which intersects the Mars orbital.
Cartesian coordinates of P1 (x,y)

$$x = h + lcos\phi, y = h + ksin\phi$$

Where t = time of opposition, taking 1^{st} opposition as the starting point (t=0) with center A (1,c) in polar coordinate and radius r
s = angular velocity of Mars
z = angle b/w equant spoke at t=0 to Eq point
l is a variable, distance b/w Eq and P1
For point P1 to interscent with the Mars orbital it must satisfy the equation of circle

$$(x - cos(c))^2 + (y - sin(c))^2 = r^2 \quad (1)$$

From (1) and Point P1,

$$l^2 + b`l + c = 0 \quad (2)$$

Where,

$$b` = 2 * (h`cos\phi + k`cos\phi)$$
$$c` = h`^2 + k`^2 - r^2$$
$$h` = h - cos\phi, k` = k - cos\phi$$

From (2), we obtained P1(x,y), following which we can determine the angle of P1 from Sun (0,0) using

$$\theta = \frac{y}{x}$$

From oppositions data, we already know longitude. Hence, Error = $\theta - longitude$
Will repeat this procedure for all the oppositions data, to get error corresponding to all 12 oppositions.
Taking the parameters (e1,e2,c,z,s,r) as input with an inital guess, and then using grid search to minimize the range of search in subsequent iterations to arrive at the most optimal point. Figure 1 below shows all the Mars orbital that can be constructed from oppositions data with some Error. It depicts equant spoke 0 and 1, similarly all 12 spokes, 0...11 can be constructed, all having some or no errors.

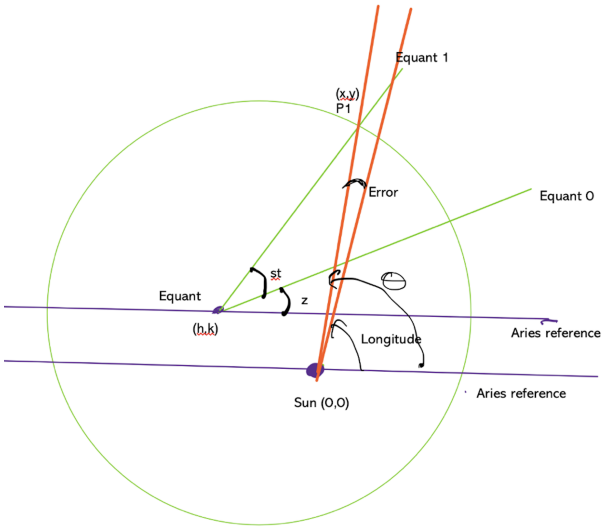

Figure 1: Mars Orbital constructed from oppositions data, with some error

2 Code Architecture

Code Follows the following algorithm

- Opposition Data is loaded. From the data we obtained Heliocentric Longitude by using the following formula - ZodiacIndex*30 + Degree + Minute/60 + Second/3600 (degrees).
- We calculate the time difference between all the oppositions data and we store the time and longitude data in an array named as oppositions
- Created a global parameter "paras", this will store the updated values of all the parameters in list format. Parameters used in subsequent questions will use the parameter values stored in "paras" as an initial guess, and will update this global parameter on their completion.
- Have created a function "minimize" which calculates the errors corresponding all the oppositions data using the procedure mentioned in the Procedure section.
- For Question 1, have made an initial guess for all the parameters. This initial guess is obtained by multiple runs with different parameter ranges and with different step size.
- Each function corresponding to all the questions, they take the initial guess of parameters from the results obtained from the last question. Then they create an array with a particular step size. Now, an iterable list is created to get all the combinations among these parameter arrays.
- The iterable combination of all parameter values "comb_list" along with Oppositions data is send to "minimize" function to obtain error list (Errors in model fitting for all oppositions data) and the best parameter. Now each function returns the data required from that particular function after updating the "paras" global variable for future use.

3 Experimentation

Have started the experimentation with an initial guess, having some prior knowledge of all the parameters. Initialized with a large step size to obtain multiple minimum points. With subsequent runs, step size value is decreased to converge towards these multiple supposedly optimal points and find the global minima amongst those. After some runs, have obtained a maximum error(amongst all oppostions data) of **18 minutes (0.30 degrees)**. The figure provided below gives the number of iterations took to obtain the corresponding maximum error and the parameters for which we obtain this error value.
Since obtained the least error on 8^{th} row in the below, have froze the python script to search in its neighbourhood.

		Inital Parameters guess					
#Iterations	Maximum Error (In Degrees)	c	s	z	r	e1	e2
2450k	7.200	170.000	0.524	54.000	9.000	1.400	120.000
1800k	1.370	170.500	0.724	59.000	9.500	2.900	119.000
1300k	1.200	179.000	0.724	59.000	9.500	2.900	119.000
1300k	4.810	179.790	0.724	64.800	9.799	3.200	119.830
1100k	7.660	180.200	0.524	53.500	9.300	180.200	1.700
2450k	9.262	142.900	0.524	55.000	9.400	1.300	145.000
1200k	0.304	149.150	0.524	56.100	8.850	1.650	92.650
960k	**0.301**	**149.240**	**0.524**	**56.100**	**8.850**	**1.650**	**92.670**

Figure 2: Experiments by varying step sizes, decreasing when near a local minima, increasing the step size when locating some optimal point

4 Conclusion

Obtained 18 minutes of the maximum error(amongst all the oppositions data). This module provided a good practice on python programming and also obtaining error function was a good exercise. Finding the global minima is a challenge, given the exponential rise of the number of iterations, is interesting.